Convolutional Neural Network vs Traditional Methods for Offline Recognition of Handwritten Digits

Edwin A. Enriquez[1], Nelly Gordillo[1] Luis M. Bergasa[2], Eduardo Romera[2], and Carlos G. Huélamo[2]

[1] Department of Electrical and Computer Engineering, Institute of Engineering and Technology, Autonomous University of Ciudad Juarez, Juarez, Chihuahua, Mexico, al150671@alumnos.uacj.mx, nelly.gordillo@uacj.mx
[2] Electronics Department, University of Alcalá (UAH), Alcalá de Henares, Spain, luism.bergasa@uah.es, eduardo.romera@depeca.uah.es, carlos.gomezh@edu.uah.es

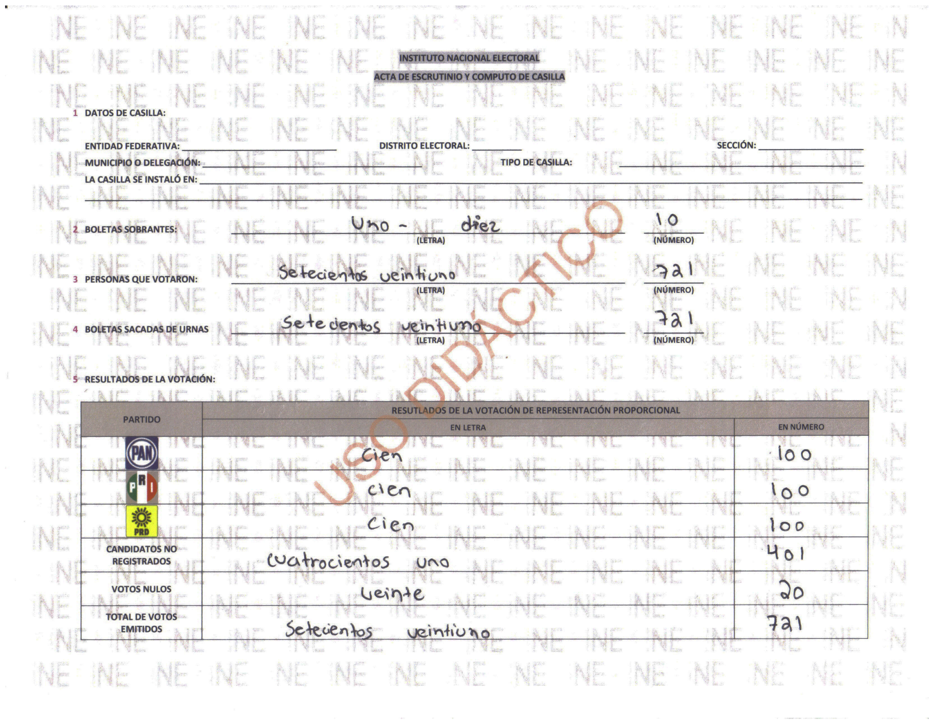
Abstract. This paper compares Convolutional Neural Networks vs traditional features extraction and classification techniques for an offline recognition of handwritten digits application. The studied classification techniques are: k-NN, Mahalanobis distance and Support Vector Machines (SVM); and the hand-designed features extraction ones are: Hu Invariant Moments, Fourier Descriptors, Projections Histograms, Horizontal Cell Projections, Local Line Fitting and Zoning. The study was conducted in a practical application as is the validation of democratic elections using ballots of electoral scrutiny with non-homogeneous background. To do that it was necessary to use different preprocessing techniques (RGB conversion to gray scale, binarization and noise reduction) as well as a segmentation stage.

Keywords: handwritten digits recognition, hand-designed features extraction, classification, CNN

1 INTRODUCTION

Offline handwritten character recognition has been an interesting topic in the fields of pattern recognition and deep learning from decades. As handwritten characters are unconstrained and topologically diverse, its recognition with pure offline information is not trivial. In an offline handwritten recognition system the writing is usually captured optically by a scanner and provided as an input image. Its goal is to digitize the handwriting by converting it into machine readable ASCII format [1]. Online methods have shown to be superior to offline ones in recognizing handwritten characters, but as several applications require optical recognition, offline methods continue being an active research topic.

In the electoral process of most of the developing countries, the scrutiny of the votes is manually done. In consequence, this electoral phase is prone to counting errors. Fig. 1 depicts an example of a ballot corresponding to an electoral process in Mexico. This is an application where an optical handwritten digits recognizer can help to automate the counting process, minimizing the human errors. It should be noted that the electoral ballot is self-designed, therefore, it is not a predefined document where standard techniques can be used without a preprocessing and segmentation stage.


INSTITUTO NACIONAL ELECTORAL

ACTA DE ESCRUTINIO Y COMPUTO DE CASILLA

1 DATOS DE CASILLA:

ENTIDAD FEDERATIVA: ______ DISTRITO ELECTORAL: ______ SECCIÓN: ______

MUNICIPIO O DELEGACIÓN: ______ TIPO DE CASILLA: ______

LA CASILLA SE INSTALÓ EN: ______

2 BOLETAS SOBRANTES: Uno - diez (LETRA) 10 (NÚMERO)

3 PERSONAS QUE VOTARON: Setecientos veintiuno (LETRA) 721 (NÚMERO)

4 BOLETAS SACADAS DE URNAS Setecientos veintiuno (LETRA) 721 (NÚMERO)

5 RESULTADOS DE LA VOTACIÓN:

PARTIDO	RESUTLADOS DE LA VOTACIÓN DE REPRESENTACIÓN PROPORCIONAL	
	EN LETRA	EN NÚMERO
PAN	Cien	100
PRI	cien	100
PRD	cien	100
CANDIDATOS NO REGISTRADOS	cuatrocientos uno	401
VOTOS NULOS	veinte	20
TOTAL DE VOTOS EMITIDOS	Setecientos veintiuno	721

Fig. 1. Example of scrutiny ballot

In the state of the art there are many papers that have tackled this problem by using different traditional methods [2] [3] [4]. A survey of these methods can be found in [1]. In the last years, Convolutional Neural Networks (CNNs) [5] have received a lot of attention due to its ability to recognize handwritten digits, reaching a moderate good performance [6] [7] [**?**].

This paper compares some traditional methods versus a CNN for offline handwritten digits recognition in a real application of electoral ballot counting in Mexico. We have used the CHARS74k and MNIST datasets for training and a particular dataset called UACJ280, formed for 6904 digits extracted from real ballots, for testing. The paper is organized as follows: chapter two presents the foundations of used traditional methods of the state of the art as well as the used CNN for our handwritten digits recognition system. Later in chapter three the different architectures tested are detailed. Finally, chapter four presents some comparative experimental results obtained for the tested architectures and chapter five describes the conclusions of this work.

2 FOUNDATIONS

2.1 Traditional Methods

Traditional handwriting recognition methods normally have four stages named: preprocessing, segmentation, feature extraction and classification. Hereafter, we sketch the foundations of the different methods used for each stage in our application.

Preprocessing In this stage we apply a series of operations over the scanned image to improve its quality for the next segmentation stage. Firstly, the color image is resized and converted to a grayscale image, and after that the following techniques are applied: binarization and noise reduction.

Segmentation A decomposition of the whole image into multiple segments is carried out in this stage. We use a simple but efficient simple pixel counting method for segmentation. The binarized image is scanned from left to right and from top to bottom.Then, an horizontal and vertical projection profiles are obtained. From these profiles lines and digits can be easily segmented by cropping the image at the minimum of the horizontal and vertical profiles respectively [1].

Feature Extraction This stage finds out some representative hand-designed features for each digit useful for the next classification stage. The different used methods are shown below.

A very established shape features are the **Hu invariants moments** [8]. The 2-D moment of $(p+q)$ order of a $f(x,y)$ image, where the point x defines the columns and y the rows, is defined by:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \tag{1}$$

For $p,q = 0,1,2,...$, where the summations move through the spatial (x,y) coordinates of the image. Its corresponding central moment is:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y) \tag{2}$$

where:

$$\bar{x} = \frac{m_{10}}{m_{00}} \tag{3}$$

$$\bar{y} = \frac{m_{01}}{m_{00}} \tag{4}$$

Being $(\bar{x},\bar{y})$ the centroid of the digit. The normalized moments of order $(p+q)$ are:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{5}$$

where:

$$\gamma = \frac{p+q}{2} + 1 \tag{6}$$

the γ function is only defined to $p+q = 2,3,...$ The set of seven Hu invariant moments [9] are insensitive to translation, rotation and scale. The invariant moments are expressed as follows:

$$\theta_1 = \eta_{20} + \eta_{02} \tag{7}$$

$$\theta_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{8}$$

$$\theta_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{9}$$

$$\theta_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{10}$$

$$\theta_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{11}$$

$$\theta_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{12}$$

$$\theta_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{13}$$

Another method is the **Fourier Descriptors** [10][8], which use the coordinates of the image contours in the form of complex number $s(k)$ expressed as:

$$s(k) = x(k) + jy(k) \tag{14}$$

where $k = 0,1,...,K-1$. To obtain the Fourier Descriptors $a(u)$ we use the Discrete Fourier Transform defined by:

$$a(u) = \sum_{k=0}^{K-1} s(k)\exp(-\frac{i2\pi uk}{K}) \tag{15}$$

On other side, [10] shows details of the **Projection Histograms** method expressed by:

$$P_{hor}(y_0) = \sum_{x=1}^{N} f(x,y_0) to 1 \leq y_0 \leq m \tag{16}$$

$$P_{ver}(x_0) = \sum_{y=1}^{M} f(x_0,y) to 1 \leq x_0 \leq n \tag{17}$$

where m and n are height and width of the image respectively.

By Zahid [11], the **Horizontal Cell Projections** method performs a partition of the character image in k regions. For the case of a horizontal cell projection, the characteristic vector of the r-th cell, corresponding to a character of m x n pixels, is expressed as $P_r = \langle P_1, P_2, ..., P_M \rangle$ in which:

$$P_i = \bigcup_{j=1}^{n/k} f(i, \frac{n(r-1)}{k} + j) \tag{18}$$

The characteristic vector of the image will be:

$$V = P_1 \cup P_2 \cup \ldots P_k \tag{19}$$

A method proposed by Pérez et al. [12] is the **Local Line Fitting**. It consists of dividing the input image (character) into k cells (meshing) and obtaining three characteristics for each of them. Firstly, the number of pixels in 1's for each cell i belonging to a character is calculated (n_i) and then its value is divided between the total number of pixels that compose the character (N), that is:

$$f_{i1} = \frac{n_i}{N} \tag{20}$$

The second and third attribute consists of performing a orthogonal regression for each cell to obtain an estimated straight line for a set of data of the form $y = a_i + b_i x$, where a_i is the intersection with y axis and b_i is the slope of the straight line. The attributes f_{i2} and f_{i3} are:

$$f_{i2} = \frac{2b_i}{1+b_i^2} \tag{21}$$

$$f_{i3} = \frac{1-b_i^2}{1+b_i^2} \tag{22}$$

In [13] and [14] the **Zoning** method is presented, which divides the grayscale or binarized character in a m x n mesh. Each cell counts the number of pixels with 1 value (or the grayscale average) providing a vector of features m x n length.

Clasiffiers

- *k*-**Nearest Neighbor (k-NN):** This technique is one of the best known for classification. The input consists of the k closest training examples in the feature space. The output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.When $k = 1$ k-NN becomes in a metric Euclidean distance classifier.
- **Mahalanobis distance:** This gives a measure of the distance between a features vector (x_{test}) and a distribution given by its mean vector (m_l), which represents the centroid of each class previously calculated in the training phase. This distance is similar to the Euclidean distance but in this case it uses a covariance matrix for each class. The covariance S is a quadratic matrix n x n length, whose main diagonal contains the variances of each feature ($S^2_{ij} \mid i = j$) and the rest of elements $i \neq j$ define the respective covariances values among features ($S^2_{ij} \mid i \neq j$)
- **Support Vector Machines (SVM):** It has been extensively used due to its easy training and the high performance obtained in classification. SVM is based on the choice of a kernel function and the penalty parameter C. The kernel maps nonlinear samples into a higher dimensional space and handles the nonlinear relations between class labels and features. SVM is widely used in classic machine learning where feature extraction phase is handmade and classification phase is learned from data in a previous training process.

2.2 Convolutional Neural Networks

CNNs are a class of deep, feed-forward artificial neural networks that has successfully been applied to image processing. They differ of traditional methods in that both the features extraction phase and the classification one are carried out in a previous deep learning process. CNNs use relatively little pre-processing compared to traditional image classification algorithms. This means that the CNN learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is its major advantage. CNN is a multi-layer network alternating operations of convolution and sub-sampling (features extraction) as well as full connexion and Gaussian connexion (classification). Each convolutional layer apply a convolution operation to the input image, passing the result to the next layer. Each convolutional neuron processes data only for its receptive field. Many CNN models can be found in the literature for classification: AlexNet [15], VGG [16], Le-Net [17], etc.

In this project we use a Le-Net-5 architecture because, according to [17], it is suitable for handwriting recognition. We keep the same pre-processing and segmentation stages that in the traditional methods but the features extraction and classification stages are carried out through this CNN.

3 SYSTEM ARCHITECTURE

This section shows the main characteristics of the implemented architectures according to the different stages explained before.

3.1 Pre-processing

The images to analyze are the areas where the counts of votes are located, as we depicts in the Fig. 2. As you can see, these areas are boxes present in the electoral scrutiny form to be fill and they have non homogeneous background because it usually includes an organization in charge of the process seal and watermarks. Pre-processing includes an image size reduction using bicubic interpolation, a conversion of RGB image to grayscale, a binarization and some noise reduction operations.

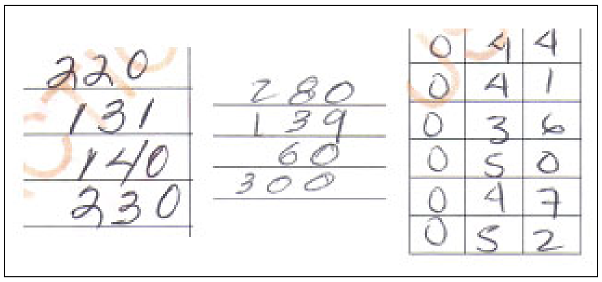


Fig. 2. Some images used

- **Binarization:** Consists of converting a grayscale image with pixel values between 0 and 255 to a binary image with [0,1] values using Otsu algorithm adaptive threshold [18]. In this case 0 indicates background pixels (black) while 1 indicates foreground pixels (white).
- **Noise reduction:** This operation removes any unwanted bit-patterns, which are noise for the digits segmentation. To do that we apply morphological operators and small blobs removing. In a first instance an erosion of the binary image is carried out to filter out the digits to be recognized. Subsequently a dilatation is made to consolidate the remain blobs [10]. Finally, we eliminate the objects whose area is lower that 20 percent of the largest blob (digit).

3.2 Segmentation

For the segmentation we apply a two steps process: lines segmentation and characters segmentation, based on pixel counting technique [1]. In the first step, a horizontal projection of the pre-processed binary image is made. We perform a row-by-row scan to detect a row of the image which projection value is zero. This means that the end of an information line is detected and therefore is segmented. In the second step a vertical projection is applied. A column-by-column scan is performed looking for zero-values. These indicate the transitions among characters and are used to segment them. This process is repeated until the whole digits of the image are segmented.

3.3 Features extraction using traditional methods

At this stage, some hand-made parameters, which provide an accurate representation of the digits to the classification stage, are found out. The used techniques are shown below.

- **Hu Invariants Moments:** We use the seven invariant moments described in previous section, applying equations (7) to (13).
- **Fourier Descriptors:** From the characteristics obtained by equation (15), only the first thirty-two descriptors of each digit are taking into account for the classification stage. This value was chosen in an experimental way.
- **Projection Histograms:** We use features calculated from equations (16) and (17), separately and jointly, for classification. More details will be given in next section.
- **Horizontal Cell Projections:** For features extraction the digit patch is divided into three up to eight cells to obtain the horizontal and vertical projections. In the same way as the ordinary projections seen in the previous method, a separate comparison is made for the obtained horizontal and vertical projections.
- **Local Line Fitting:** Each digit patch is divided into a mesh of n x m cells. For each cell the features calculated by equations (20) to (22) are extracted with the difference that a regression is done by the method of least squares to obtain the slope. The values of n and m are the same which comprise $n, m = 3, 4, ..., 8$.
- **Zoning:** This method, comparing the previous one, only divides the digit into a mesh of n x m cells for the same values seen previously and makes for each cell a count of the pixels belonging to the digit creating a vector of characteristics n x m length.

3.4 Classification using traditional methods

In this stage each digit patch is assigned to a class using some traditional methods based on distance.

- ***K*-Nearest Neighbor:** This method calculate the distance of an unknown extracted features vector to the K nearest neighbors and classify it as belonging to the class to most of the neighbors belong. K was chosen in an experimental way to 15. There is a training phase consisting in obtaining a k x l matrix, where k are the chosen features and l is the number of characters used.
- **Mahalanobis distance:** It is similar to the previous method but in this case the used distance metric takes into account the covariance of each class. In the previous training process the mean and covariance matrix must be calculated for each digit class.
- **SVM:** In this case we use a multi-class SVM classifier. We made several tests and the best results were obtained for an linear kernel and a penalty parameter C=1.

3.5 CNN for features extraction and classification

For the features extraction and classification stages, we have used a CNN based on LeNet-5 [17]. The architecture of our network is as follows:

- Layer S1 is an input patch corresponding to a digit of size 28 x 28 pixels.
- Layer C1 is the first convolutional layer with 20 feature maps of size 28 x 28. Each unit is connected to a 5 x 5 neighborhood of the input layer S1.
- Layer S2 is the second sub-sampling layer with 20 feature maps of size 14 x 14. Each unit is connected to a 2 x 2 neighborhood in the corresponding feature map in layer C1.
- Layer C3 is the second convolutional layer. This contains 50 feature maps of size 8 x 8. Each unit is connected to a 5 x 5 neighborhood in the corresponding feature map in layer S2.
- Layer S4 is the third sub-sampling layer containing 50 feature maps of 4 x 4 where each unit is connected to a 2 x 2 neighborhood of the layer C3.
- Layer C5 is the third convolutional layer. This contains 500 feature maps of size 1 x 1. Each feature map is connected to all 50 feature maps of layer S4.We have added a ReLu activation function that adds nonlinearity to the network.
- Layer F6 contains 60 units and it is fully connected to layer C5.
- Layer F7 (the output layer) is composed of Euclidean RBF units and it is a fully connected to layer F6. We have used a softmax layer, in which it shows the probabilities of each class, for error calculation.

4 RESULTS

4.1 Handwritten Databases

For this work, the used training databases were CHARS74K [19] and MNIST [20]. The first database contains a total of 3,410 elements between the ten digits of 0-9, twenty-six characters from a to z in lowercase and another group of twenty-six characters but in uppercase. Each character has fifty-five types of writing. It should be noted that for the training in the algorithms implemented only the 0-9 digits were used, giving a total training of 550 images resized to a 28 x 28 resolution.

On the other hand, for the MNIST database, only the 60,000 digit training database was used, which was divided into two groups. The first consisted in a training group of 10,000 (MNIST1000) divided equally among the ten digits. The same was done with the second group of 54,000 elements (MNIST5400). Image dimensions were 28 x 28 for the distance algorithms and the convolutional network.

For the test phase the digits extracted from the ballots were used as a result of the pre-processing that will be detailed below. The UACJ280 database has a total of 6904 digits from 0 to 9. The dimensions for this database were the same that in the training phase.

In the following tables, classification percentages for the UACJ280 database are shown as a function of the three different training databases and for the six features extraction techniques used for each classifier (k-NN, Mahalanobis distance and SVM). Table 1 shows the highest recognition percentages for the k-NN classifier. The Hu Invariant Moments presented the percentages of classification of smaller magnitude in comparison of the other features extraction algorithms. Results do not exceed 41% of correct detection for the UACJ280 database. On the other hand, the Fourier Descriptors showed variability in the classifications for each training set, not exceeding 56%

Table 1. Classification percentages using K- Nearest Neighbor

Feature	CHARS74K	MNIST1000	MNIST5400
Hu Invariants Moments	41%	38%	40%
Fourier Descriptors	46%	53%	56%
Projections Histograms	63%	63%	66%
Cells Proyections	**82**%	91%	94%
Local Line Fitting	78%	88%	92%
Zoning	80%	**92**%	**95**%

of the total characters for the MNIST5400 database. The percentages of Projections Histograms are referring to those of horizontal type achieving recognition numbers between 63% and 66%. In the case of Cell Projections for five horizontal cells, it shows percentages higher than 80%, obtaining the highest percentage for the MNIST5400 database with more than 94% of correct detections. For the case of Local Line Fitting with a 6 x 6 grid, using the MNIST5400 as training dataset, 92% was obtained in recognizing the digits of UACJ280. Finally, for the method of Zoning, a similar behavior is shown as for the cell projections one.

Mahalanobis distance classifier (Table 2) shows different percentages and in most of the cases are lower compared to the previous classifier. CHARS74K shows the lowest percentages in recognition having about 41%. MNIST5400 is about 55% and finally MNIST5400 is about 62%. It should be noted that percentages shown in this table are obtained for the following features extractor cases: Projection Histograms uses only the horizontal type. Cell Projections applies a combined evaluation between horizontal and vertical projections of three cells. Local Line Fitting uses a 3 x 3 grid while Zoning uses a 5 x 5 grid.

For the evaluation applying SVM (Table 3) there are percentages of recognition below that expected maybe due to only a linear kernel was used. It should be noted that the percentages of classification for Hu Invariant Moments are the lowest (same characteristic in other classifiers) between 12% and 40%. In contrast, for Fourier Descriptors the percentages are between 40% and 50% between the three bases of training where 47% of the UACJ280 test elements were classified by MNIST1000. Highlight the Histograms of Horizontal Projections have 60% expected value in conjunction with the three training databases described above. Meanwhile, projections by eight cells have recognition percentages of 86% and 89% for the MNIST1000 and MNIST5400 databases, respectively. Results for CHARS74K database show a high difference of 22% compared to MNIST1000. The Local Line Fitting method using a 6 x 6 grid presents a behavior similar to the previous method, obtaining a difference of 2% between their respective

Table 2. Classification percentages using Mahalanobis distance

Feature	CHARS74K	MNIST1000	MNIST5400
Hu Invariants Moments	41%	41%	43%
Fourier Descriptors	37%	37%	37%
Projections Histograms	40%	55%	60%
Cells Proyections	30%	73%	81%
Local Line Fitting	21%	34%	57%
Zoning	**76**%	**89**%	**93**%

training bases. Finally, the extraction by zoning method for 5 x 5 grid presents 68% for CHARS74K, 85% for MNIST1000 and 87% for MNIST5400.

Table 3. Classification percentages using support vector machines

Feature	CHARS74K	MNIST1000	MNIST5400
Hu Invariants Moments	12%	39%	40%
Fourier Descriptors	40%	47%	50%
Projections Histograms	55%	61%	63%
Cells Proyections	64%	**86**%	**89**%
Local Line Fitting	66%	84%	87%
Zoning	**68**%	85%	87%

Results for the classification technique are shown in Table 4. Convolutional Neural Networks were analyzed with fifteen and twenty-five training periods for the neural network. The classification percentages obtained by this technique were the highest in comparison to the traditional classification techniques. In the case of fifteen seasons, 85% classification percentage is obtained for CHARS74K; 93% for MNIST1000 and 98% for MNIST5400, which is 5% over MNSIT1000. For the case for 25 epoch, the same trend is observed, that is, for CHARS74K and MNIST1000 datasets there is a difference of 1% over results with fifteen seasons, and for MNIST5400 the same percentage prevails.

Table 4. Classification percentages using Convolutional Neural Networks

EPOCH	CHARS74K	MNIST1000	MNIST5400
15	85%	93%	98%
25	**86**%	**94**%	**98**%

5 CONCLUSIONS

We conducted a comparative study of Convolutional Neural Networks vs. traditional methods of classification, which led to a practical level that is the classification for an offline recognition of hand written digits application as is the validation of democratic election using ballots of electoral scrutiny with non homogeneous background. We conclude that results obtained with the CNN are quite better than the obtained with traditional methods. It should be noted that all the classifiers were susceptible to the amount of training data, in other words, the larger the database to be trained, the better the classification of the test elements.

6 ACKNOWLEDGMENT

This work has been partially funded by the Spanish MINECO/FEDER through the SmartElderlyCar project (TRA2015-70501-C2-1-R), the DGT through the SERMON project (SPIP2017-02305), and from the RoboCity2030-III-CM project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos, fase III; S2013/MIT-2748), funded by Programas de actividades I+D (CAM) and cofunded by EU Structural Funds.

References

1. A. Manoj, P. Borate, P. Jain, V. Sanas, and R. Pashte, "A survey on offline handwriting recognition systems," 2016.
2. M. Parizeau, A. Lemieux, and C. Gagné, "Character recognition experiments using unipen data," in *icdar*, p. 0481, IEEE, 2001.
3. R. Ghosh, M. Ghosh, *et al.*, "An intelligent offline handwriting recognition system using evolutionary neural learning algorithm and rule based over segmented data points," *Journal of Research and Practice in Information Technology*, vol. 37, no. 1, p. 73, 2005.
4. V. Nguyen and M. Blumenstein, "Techniques for static handwriting trajectory recovery: a survey," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pp. 463–470, ACM, 2010.

5. Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
6. D. Bouchain, "Character recognition using convolutional neural networks," *Institute for Neural Information Processing*, vol. 2007, 2006.
7. F. Lauer, C. Y. Suen, and G. Bloch, "A trainable feature extractor for handwritten digit recognition," *Pattern Recognition*, vol. 40, no. 6, pp. 1816–1824, 2007.
8. R. C. Gonzalez, R. E. Woods, S. L. Eddins, *et al.*, *Digital image processing using MATLAB.*, vol. 624. Pearson-Prentice-Hall Upper Saddle River, New Jersey, 2004.
9. M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.
10. E. Cuevas, D. Zaldívar, and M. Pérez, *Procesamiento digital de imágenes con MATLAB & Simulink*. Ra-Ma, 2016.
11. M. Z. Hossain, M. A. Amin, and H. Yan, "Rapid feature extraction for optical character recognition," *arXiv preprint arXiv:1206.0238*, 2012.
12. J.-C. Perez, E. Vidal, and L. Sanchez, "Simple and effective feature extraction for optical character recognition," in *Selected Papers From the 5th Spanish Symposium on Pattern Recognition and Images Analysis: Advances in Pattern Recognition and Applications, Valencia, Spain*, 1994.
13. M. Bokser, "Omnidocument technologies," *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1066–1078, 1992.
14. O. D. Trier, A. K. Jain, T. Taxt, *et al.*, "Feature extraction methods for character recognition-a survey," *Pattern recognition*, vol. 29, no. 4, pp. 641–662, 1996.
15. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
16. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
17. A. Manoj, P. Borate, P. Jain, V. Sanas, and R. Pashte, "Offline handwriting recognition system using convolutional network," 2016.
18. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
19. T. de Campos, "The chars74k dataset: Character recognition in natural images. university of surrey. guildford, surrey, uk," 2012.
20. Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.